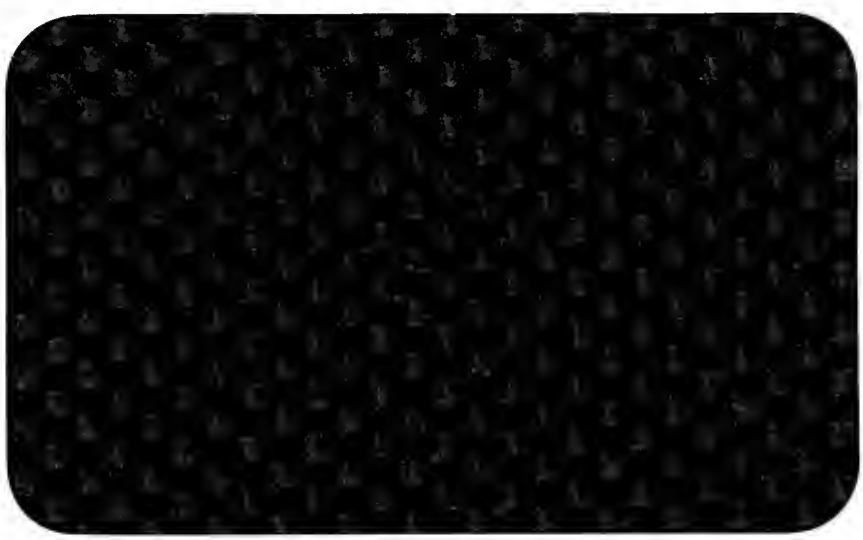
NUMERICAL SOLUTION OF A MODEL PROBLEM FROM COLLAPSE LOAD ANALYSIS*

By

Michael L. Overton

February, 1984

Report #100



NUMERICAL SOLUTION OF A MODEL PROBLEM FROM COLLAPSE LOAD ANALYSIS*

By

Michael L. Overton

February, 1984

Report #100

*This paper will appear in the Proceedings of the Sixth International Symposium on "Computing Methods in Engineering and Applied Sciences", sponsored by the Institut National de Recherche en Informatique et en Automatique (INRIA), Versailles, France, December 12-16, 1983, to be published by North-Holland (Amsterdam, 1984).

NUMERICAL SOLUTION OF A MODEL PROBLEM FROM COLLAPSE LOAD ANALYSIS

Michael L. Overton

Courant Institute of Mathematical Sciences
New York University
New York, N. Y. 10012
U.S.A.

We consider a model problem from collapse load analysis discussed recently by Strang. The analytic solution is a characteristic function with a jump discontinuity. We develop a method for solving a discretized version of the model problem, which requires the minimization of a convex piecewise differentiable function. Numerical results are presented.

INTRODUCTION

In collapse load analysis the following problem arises (see Strang [16]):

$$\min_u \int_\Omega \|\nabla u\| \tag{1.1a}$$

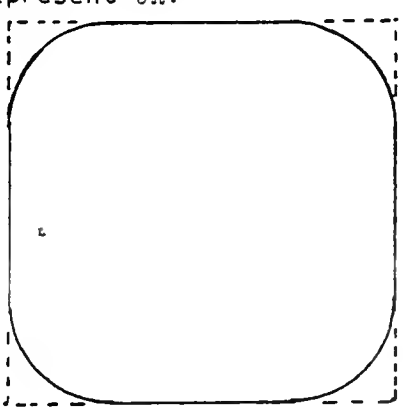
subject to the Dirichlet boundary conditions

$$u(x,y) = f(x,y) \quad \text{on} \quad \partial\Omega \tag{1.1b}$$

and the constraint

$$\int_\Omega c(x,y)\, u(x,y) = 1 \tag{1.1c}$$

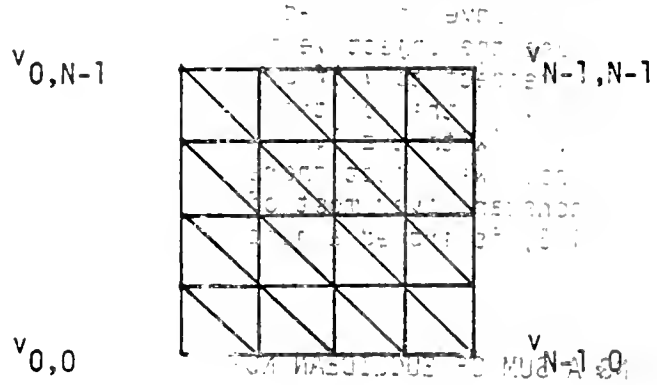
Here $\|\nabla u\|$ denotes the Euclidean norm of the gradient of $u(x,y)$, i.e. $\sqrt{u_x^2+u_y^2}$. We shall restrict Ω to be a square with side two. The functions f and c are usually smooth functions, defined on $\partial\Omega$ and Ω respectively. Sometimes the constraint (1.1c) does not appear, but this is required to obtain a nontrivial solution if, for example, $f(x,y) \equiv 0$. In order for a minimum to be achieved, the appropriate space of functions to consider is BV, the functions of bounded variation. Strang shows, using a result of Fleming [9], that when $c(x,y) \equiv 1$, $f(x,y) \equiv 0$, the solution $u(x,y)$ is a characteristic function with a jump discontinuity along a curve Γ in Ω. Specifically, $u(x,y)$ has constant positive value inside Γ and zero value outside Γ in the following figure, where Γ is the solid curve and the broken lines represent $\partial\Omega$:



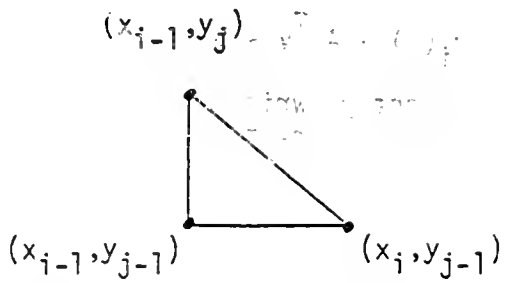
(1.2)

The curve Γ consists of four circular arcs of radius $\rho = 1/(1+\sqrt{\pi}/2) \approx 0.530$, plus parts of the sides of the square. Once it is recognized that the solution is a characteristic function, (1.1) reduces to a classical isoperimetric problem, since the quantity to be minimized reduces to measuring the length of Γ while the constraint fixes the area inside Γ. The solution therefore defines the region in the square with minimal perimeter given fixed area (maximal area given fixed perimeter).

In this paper we are concerned with obtaining the numerical solution of a discrete approximation to (1.1). Let us triangulate Ω as follows:



Let $h = 1/(N-1)$ be the mesh size, where there are N mesh points in each direction. We replace u in (1.1) by a piecewise linear finite element approximation v, obtaining a finite dimensional optimization problem. The variables are the unknown function values $v(x_i,y_j)$ at the given mesh points, $i = 0,\ldots,N-1$, $j = 0,\ldots,N-1$. Because v is piecewise linear, the norm of its gradient $v_\cdot = (v_x,v_y)^T$ on a "lower" triangle:



multiplied by the area of the triangle, is given by

$$\| r_{ij1} \| = \left\| \begin{bmatrix} \frac{h}{2}\,(v(x_i,y_{j-1}) - v(x_{i-1},y_{j-1})) \\ \frac{h}{2}\,(v(x_{i-1},y_j) - v(x_{i-1},y_{j-1})) \end{bmatrix} \right\| .$$

Let us write

$$r_{ij1} = A_{ij1}{}^T v \in \mathbb{R}^2$$

where v is the vector $\in \mathbb{R}^{N^2}$ of unknowns. The matrix A_{ij1} has dimension $N^2 \times 2$ and has only two nonzero components in each column. We use r_{ij2} and A_{ij2} for the corresponding notation for "upper" triangles. Thus the discretized problem is:

$$\min_v \sum_{i,j,k} \| r_{ijk} \| \qquad (1.3a)$$

$$(i=1,\ldots,N-1,\ j=1,\ldots,N-1,\ k=1,2)$$

subject to the boundary conditions

$$v(x_i,y_j) = f(x_i,y_j) \quad \text{for} \quad (x_i,y_j) \in \partial\Omega \tag{1.3b}$$

and the constraint

$$\frac{1}{N^2} \sum_{i,j} c(x_i,y_j)\, v(x_i,y_j) = 1 \ . \tag{1.3c}$$

Although (1.3) is a convex finite-dimensional optimization problem, it is not easily solved, because the objective function to be minimized is only piecewise differentiable with respect to v if any of the r_{ijk} equal zero. Now the residual r_{ijk} is zero precisely if the solution is constant on the corresponding triangle. The solution of (1.1) (with $c \equiv 1$, $f \equiv 0$) is constant everywhere except across Γ where it jumps. We expect therefore that the solution to the discretized problem will be constant over most of Ω but varying in some band around Γ. In this case (1.3) is indeed a nondifferentiable optimization problem.

AN ALGORITHM FOR MINIMIZING A SUM OF EUCLIDEAN NORMS

We have recently proposed [14] an algorithm for solving nondifferentiable optimization problems of this form. Somewhat more generally, it solves the convex problem:

$$\min_{v \in \mathbb{R}^n} F(v) = \sum_{i=1}^{m} \| r_i(v) \| \tag{2.1}$$

where

$$r_i(v) = A_i^T v - b_i \in \mathbb{R}^{\ell}.$$

We are changing the notation slightly, writing i where before we wrote the triple (i,j,k), and now allowing the residual r_i to be affine rather than linear in v. As before, $\|\cdot\|$ denotes the Euclidean vector norm. Linear constraints may be included without difficulty, but we omit them for convenience. The problem (2.1) dates back to Fermat and has an interesting history. In its simplest form, with $n = \ell = 2$, it asks for the point in the plane which minimizes the sum of distances between it and m given points. A more interesting version involves a weighted sum of the distances. With this variation, the solution coincides with one of the given points if the corresponding weight is large enough. This problem is also associated with the names of Steiner and Weber and is often known as the single facility location problem. The multifacility location problem arises when more than one point in the plane is to be determined, i.e. $n > \ell = 2$. The weighted sum of norms to be minimized then involves the distances between each pair of unknowns as well as the distances between the unknown and fixed points. This particular problem is discussed in more detail by Calamai and Conn [2,3,4], who give an algorithm closely related to ours.

The algorithm as described in [14] is intended to solve (2.1) when the matrices $\{A_i\}$ are dense and n is not too large; a full rank condition is also required. None of these conditions holds in (1.3). The purpose of this paper is to explain how to adapt the algorithm to solve (1.3), and to report on the advantages and difficulties of such an approach.

Notation. Using ∇ to denote gradient with respect to v, we have:

$$g_i(v) \equiv \nabla \| r_i(v) \| = \frac{1}{\| r_i(v) \|} A_i r_i(v)$$

$$G_i(v) \equiv \nabla^2 \|r_i(v)\| = \frac{1}{\|r_i(v)\|} A_i A_i^T - \frac{1}{\|r_i(v)\|^3} A_i r_i(v) r_i(v)^T A_i^T$$

provided that $\|r_i(v)\| \neq 0$. Note that the Hessian term $G_i(v)$ is unbounded as $\|r_i(v)\| \to 0$, but that the gradient term $g_i(v)$ remains bounded, although it is of course discontinuous at v if $r_i(v) = 0$. Now let us define

$$g(v) = \sum_{\|r_i(v)\| \neq 0} g_i(v)$$

and

$$G(v) = \sum_{\|r_i(v)\| \neq 0} G_i(v) .$$

If the gradient and Hessian of F(v) are defined it is clear that they are given by g(v) and G(v) respectively. The discontinuities in the gradient and consequent unboundedness in the Hessian are what cause the difficulty in solving (2.1).

In order to be able to handle the discontinuities consider the following idea. Define the <u>active set</u> at a point v as

$$J(v) = \{i \mid \|r_i(v)\| = 0\}, \tag{2.2}$$

i.e. those indices associated with zero residuals. We refer to r_i as an active residual or active term at v if $i \in J(v)$. The idea of the algorithm is to project the objective function F(v) into the linear manifold where zero residuals remain unchanged. In this space we see that F(v) is locally continuously differentiable, since discontinuities occur only in directions crossing the manifold. To make this precise let

$$\hat{A} = \hat{A}(v) = [A_{i_1} A_{i_2} \ldots] \quad \text{where} \quad J = \{i_1, i_2, \ldots\}$$

i.e. the matrix whose columns are the coefficient matrices of the active residuals. Let $Z = Z(v)$ be a matrix with maximal column rank such that $\hat{A}^T Z = 0$, i.e. such that the columns of Z span the null space of $\hat{A}^T$. The significance of the matrix Z is that

$$r_i(v+p) = r_i(v) = 0 \quad \text{if} \quad i \in J(v) \quad \text{and} \quad p \in R(Z) ,$$

the range space of Z. Now for any v consider the matrix Z and define F restricted to the space $v \oplus R(Z)$ as

$$F_Z(p_Z) = F(v + Zp_Z).$$

Consider the gradient and Hessian of F_Z, which will be called respectively the projected gradient and projected Hessian. Because the active terms are fixed for all p_Z, we have

$$\nabla F_Z(p_Z) = Z^T g(v + Zp_Z)$$

$$\nabla^2 F_Z(p_Z) = Z^T G(v + Zp_Z) Z .$$

Thus the projected gradient and projected Hessian are defined and differentiable in a neighborhood of v regardless of whether any residuals are zero.

<u>Optimality conditions</u>. Let us consider conditions for a point v to be a solution to (2.1), making use of the above notation. Clearly a necessary condition for v to be a solution is that the projected gradient is zero, since otherwise F could be decreased along a direction in $R(Z(v))$. Thus for optimality we require

$$Z^T g(v) = 0 . \tag{2.3}$$

By the definition of Z an equivalent condition is

$$g(x) = \sum_{i \in J(v)} A_i t_i \tag{2.4}$$

for some vectors $t_i \in R^{\ell}$ for each $i \in J(v)$. We call the $\{t_i\}$ Lagrange vectors. By considering the change in F along directions not in $R(Z)$, it can be shown that a necessary and sufficient condition for v to solve (2.1) is that (2.4) hold for some vectors $\{t_i\}$ satisfying

$$\|t_i\| \leq 1 \quad \text{for all} \quad i \in J(v) \tag{2.5}$$

(see [12], [14], [19]).

The algorithm. The basic idea of the algorithm is as follows. Given a point $v^{(0)}$ with an active set $J(v^{(0)})$, we proceed to minimize the function restricted to $v^{(0)} \oplus R(Z(v^{(0)}))$, where the zero residuals remain unchanged. At the kth step of the iteration, a direction of search is computed by solving the projected Newton system of equations, utilizing the projected gradient $Z^T g$ and projected Hessian $Z^T G Z$. A special line search is used to obtain the next iterate $v^{(k+1)}$. During the course of this iteration we may reduce other residuals to zero, in which case they are added to the active set and the restricting manifold is reduced in dimension. Once the projected gradient $Z^T g$ is sufficiently small, the Lagrange vectors $\{t_i\}$ should be computed. These are uniquely defined if $\hat{A}$ has full column rank. If $\|t_i\| \leq 1$ for all $i \in J(v)$, the procedure terminates. If $\|t_j\| > 1$ for some j, the j is deleted from the active set J, the dimension of the manifold is increased, and the process is continued.

The details of the algorithm may be found in [14]. Here we outline the main steps which must be performed to obtain a new iterate $v^{(k+1)}$ from $v^{(k)}$. It is assumed that $\hat{A}$ has full column rank and that matrix factorizations are practical; such is not the case for (1.3).

Choose active set, compute Z, and make almost active terms exactly active if necessary. (2.6)

Find the active set J and the matrix $\hat{A}$ at $v^{(k)}$. Compute Z from the QR factorization of $\hat{A}$. If some residuals are almost active, compute a point $\tilde{v}$ which makes these terms exactly active. If $F(\tilde{v}) < F(v^{(k)})$, then replace $v^{(k)}$ by $\tilde{v}$ and restart this step; otherwise reject these terms as inactive.

Check optimality (2.7)

If the projected gradient norm $\|Z^T g\|$ is small, compute the Lagrange vectors $\{t_i\}$ using the QR factorization of $\hat{A}$. If $\|t_i\| \leq 1$ for all $i \in J$ then stop; solution has been approximated. Otherwise delete an appropriate term from J and set p to a projected steepest descent direction.

Solve projected Newton system (2.8)

If $\|Z^T g\|$ is not small, solve

$$(Z^T G Z) p_Z = -Z^T g$$

using a Choleski factorization and set the direction of search $p = Z p_Z$.

Line search (2.9)

Use a special line search, which takes account of the form of F, to obtain

a point $v^{(k+1)} = v^{(k)} + \alpha p$, with $F(v^{(k+1)}) < F(v^{(k)})$.

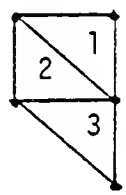
One interesting aspect of the algorithm is illustrated by considering the weighted Fermat problem:

$$\min_{v \in \mathbb{R}^2} \|v - \begin{bmatrix} -1 \\ 0 \end{bmatrix}\| + \omega\|v - \begin{bmatrix} 0 \\ 1 \end{bmatrix}\| + \|v - \begin{bmatrix} 1 \\ 0 \end{bmatrix}\| \quad . \tag{2.10}$$

Let the initial iterate be given by, say, $v^{(0)} = [3,2]^T$. We have $J(v_0) = \emptyset$ and $Z(v_0) = I$. If $\omega = 1$, the solution v^* is a point with $J(v^*) = \emptyset$, and the iterates generated by the algorithm converge quadratically to v^* because of standard properties of Newton's method. Now suppose that $\omega = 2$. It is easily verified that the solution v^* is $[0,1]^T$, with J containing one index. Now there is no obvious reason to suspect that the convergence of $v^{(k)}$ to v^* will be rapid, since F is not differentiable at v^* and the usual quadratic convergence property does not apply. In fact the Hessian $\nabla^2 F$ is unbounded in any neighborhood of v^* and undefined at v^*. It is the case, however, that this ill-conditioning of $\nabla^2 F$, combined with the effect of the special line search algorithm, actually causes quadratic convergence to v^*. Once the convergence is recognized, the exact step to v^* may be taken by (2.6). Since the new restricting manifold has zero dimension, the Lagrange vector is then computed by (2.7), and the algorithm terminates. This quadratic convergence result is similar in flavor to the well known cubic convergence of the Rayleigh quotient iteration to find an eigenvector of a symmetric matrix. In both cases the superlinear convergence is caused by the fact that the ill-conditioning of the matrices is in precisely the desired direction. For further details see [14].

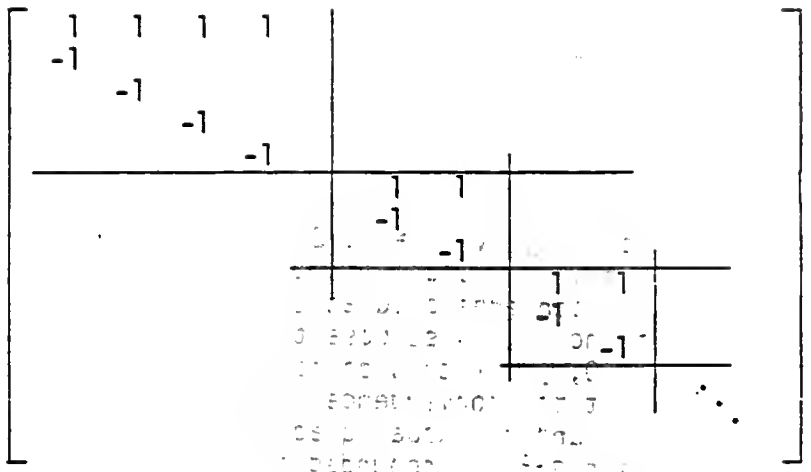
ADAPTING THE ALGORITHM TO SOLVE THE MODEL PROBLEM

Let us now consider how to adapt the algorithm of Section 2 to solve (1.3). It is convenient to continue using the notation of Section 2, i.e. writing A_i for A_{ijk}, n for N^2, etc. The first question is how to represent the active set and the matrix $\hat{A}$. First note that $\hat{A}$ is certain to be rank deficient. This is easily seen by considering the following example:



(3.1)

If the solution v is constant across triangles 1 and 3, then it must also be constant across triangle 2. Thus the three residuals being zero are not independent properties. In general, the independent columns of $\hat{A}$ can be written in the form

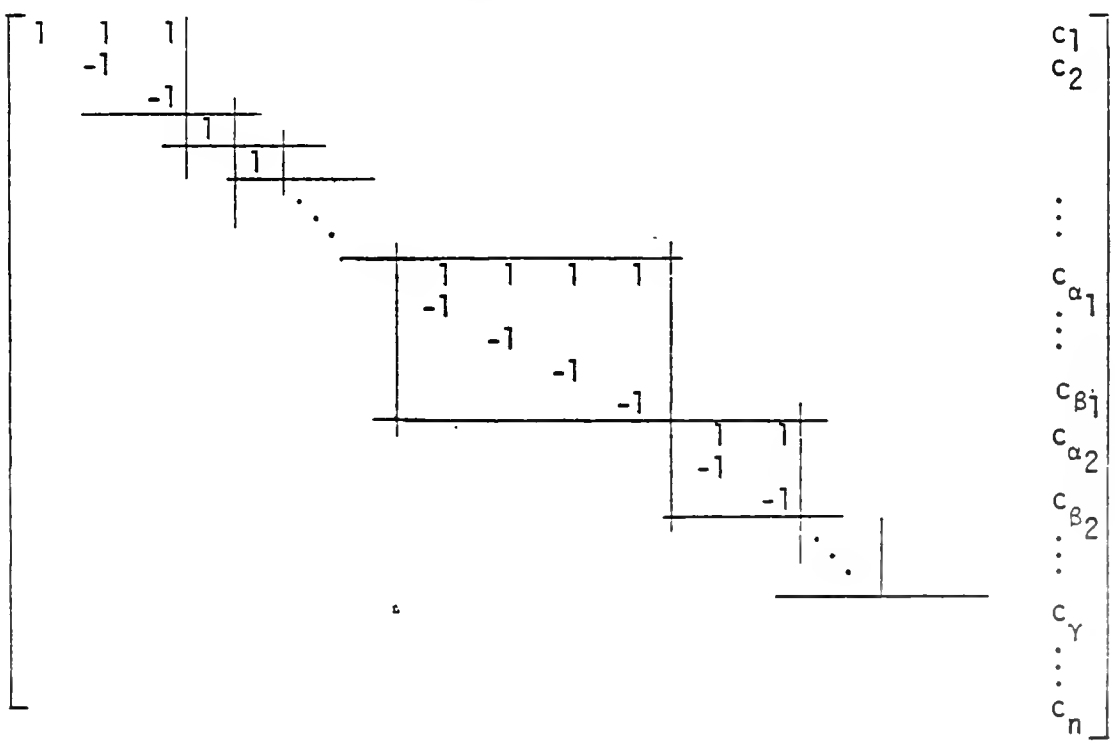


(3.2)

Recall that $\hat{A}^T p = 0$ is the equation which defines directions along which the active set does not change (zero residuals remain zero). Thus the first diagonal block of (3.2) corresponds to a region such as (3.1), where five function values $v(x_j, y_j)$ are constrained to retain the same common value. This common value, is, however, free to change. The other blocks of (3.2) correspond to other regions in Ω with constant v.

Now let us add columns to (3.2) to account for the constraints, since any direction of search p must also retain feasibility. Some of the regions with constant solution may include one or more boundary points, and hence the common function value must be fixed. Thus one column should be added to the corresponding block, making it square. Other blocks corresponding to regions away from the boundary remain unchanged. Boundary points not in any of the constant regions form their own 1×1 blocks. Let us re-order the blocks so that those corresponding to regions including boundary points appear first. The last rows, which contain no diagonal blocks, correspond to free interior points. Finally, one column corresponding to the constraint (1.3c) must be added. The matrix thus has the form:

$\tilde{A} =$



Storing and factoring $\tilde{A}$ would be prohibitively expensive. Fortunately, this is not necessary. A full rank matrix Z which spans $N(\tilde{A}^t)$ can be written in the following form:

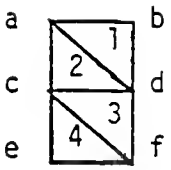
$$Z = \begin{array}{c|c|c} & 0 & \\ \hline 1 & & \\ 1 & & \\ 1 & & \\ 1 & & \\ 1 & & \\ \hline & 1 & \\ & 1 & \\ & 1 & \\ \hline & & \\ \hline d_1 & d_2 & 0 \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ (\text{row } \alpha_1) \\ \\ \\ \\ \\ (\text{row } \alpha_2) \\ \\ \\ \\ (\text{row } \gamma) \end{array}$$

$$\text{where } d_i = -(c_{\alpha_i} + \cdots + c_{\beta_i})/c_n . \qquad (3.3)$$

Note that Z is not orthogonal, which would be desirable for numerical stability. However, this is not critical. We must however ensure that we do not divide by a small number in (3.3). To avoid this, the largest of the values $|c_\gamma|,\ldots,|c_n|$ is used, with the rows interchanged accordingly. (Strictly speaking, there might be no single free interior points, i.e. $\gamma > n$, but in this unlikely case the definition of Z can be changed to have two or more dense rows instead of one.)

The best way to represent the active set and the matrices $\tilde{A}$ and Z seems to be as follows, using a linked list structure. One linked list is maintained for every block in A, including the null blocks corresponding to free interior points. These lists are of two kinds: fixed lists, corresponding to square blocks in $\tilde{A}$, and variable lists corresponding to other blocks. Thus each fixed list contains points whose common function value is fixed by the boundary conditions, and each variable list contains points whose common function value may vary. Initially, if $v^{(0)}$ has no constant regions, each boundary point is put in its own fixed list and each interior point is put in its own variable list. (Neumann conditions could also be handled by putting two adjacent points from each segment orthogonal to the boundary in their own variable list.) We also maintain a table which specifies the list containing each point.

We now focus on the best ways to carry out steps (2.6) through (2.9) of the algorithm, using the data structure just described. Let us first consider (2.6). At the beginning of each iteration, the list structure has a form which defines the values of J (and Z) at the previous iteration. It is now first necessary to add to J any indices corresponding to residuals which became "exactly" zero during the line search. This is done by merging the appropriate lists. For example, consider



(3.4)

supposing that at the previous iteration there was a zero residual across triangle 1 only, and that the values of v at points b, d, f are fixed by the Dirichlet boundary conditions. Reflecting this, there would be one fixed list specifying the values of v at (a,b,d), a second fixed list specifying the value of v at f, and two variable lists reflecting the free values at c and e. Now suppose that in the subsequent line search, residuals were reduced to zero across triangles 2 and 3 (this assumes that the Dirichlet data at d and f are the same, and involves a change only to the value at c). The two fixed lists and the variable list for c would all be merged, creating one fixed list and reflecting the new definitions of J and Z. The merge operation is trivial as it simply requires appending one linked list to another.

Step (2.6) must also consider forcing residuals which are "almost" zero to become "exactly" zero. This is a more complicated operation for two reasons. First, the current list structure must be saved, since the new vector $\tilde{v}$, for which the relevant residuals are exactly zero, may have a higher value of F than the current iterate, in which case it must be discarded. Second, the vector $\tilde{v}$ must be chosen to satisfy the constraint (1.3c). When two variable lists are merged, with two slightly different common values of v, the new variable common value can be chosen to ensure that (1.3c) is satisfied, provided that $c(x_j,y_j)$ is a strictly positive function on the whole mesh. This is the case for Strang's model problem where $c \equiv 1$. When a variable and a fixed list are merged, the variable common value must be changed to the fixed common value, potentially violating (1.3c). The simplest way to deal with this situation is to perform all the merge operations, and then restore feasibility by scaling all the variable values by a constant factor. Again, it is easy to see that this is possible provided c is a positive function. The merging of two fixed lists is not permitted when the residual is not "exactly" zero, i.e. within machine accuracy.

It is of some interest to note that in practice, many residuals are reduced exactly to zero by the line search. This contrasts with the situation for the single and multi-facility location problems, e.g. (2.10) with $\omega = 2$, where the convergence of the residual to zero is an iterative process with quadratic convergence. The reason for the different behavior here is not the rank deficiency of $\hat{A}$ per se, since this is also present in a different form for the multifacility location problem. Consider again the example (3.4). Given any $p \in R(Z)$, the only component relevant to the proposition of incorporating triangle 2 into the constant region of triangle 1 is the change in the value of v at c, or, if (a,b,d) are in a variable list, the changes of the value at c and the value at (a,b,d). Thus it is normally possible to pick a steplength α which will make $v + \alpha p$ constant across triangles 1 and 2 (when (a,b,d) are in a variable list, it is just a matter of weighting the two values correctly). In other words, for almost any $p \in R(Z)$, the extra residual can be reduced to zero along p. In the weighted Fermat problem (2.10), with $\omega = 2$, almost no direction points from a given v in the plane to the solution v^*, the only point where the relevant residual is zero.

Let us now consider the question of solving the projected Newton system (2.8). It is clearly best to use an iterative method to solve this linear system, so that it is not necessary to form or store Z^TGZ. Instead, a subroutine is required to multiply Z^TGZ onto any vector y. This can be accomplished by first multiplying by Z, then by G, and then by Z^T. Thus subroutines are required to multiply any vector y by G, Z and Z^T respectively. The first uses:

$$Gy = \sum_{i \notin J} \frac{1}{\|r_i\|} A_i \left(A_i^T y - \frac{(r_i^T A_i^T y)}{\|r_i\|^2} r_i\right)$$

where the terms corresponding to active residuals are omitted as explained in Section 2. The subroutines which multiply a vector y by Z and Z^T use the list structure, which represents Z, directly. These are also required to compute the vectors $Z^T g$ and Zp_Z. Each of the three subroutines requires O(n) operations for the matrix-vector multiplication.

Following [7], we do not usually carry out the iterative solution of the linear system to full accuracy. Let us use the conjugate gradient method to solve the system, denoting its iterates by w_k; the relevant formulas may be found in [7], with $w_0 = 0$. We terminate the iteration, setting $p_Z = w_k$, if $\|Z^T G Z w_k + Z^T g\| \leq \text{CGTOL} \|Z^T g\|$, but with a maximum of $k \leq \text{CGMAX}$ imposed unless $\|Z^T g\| \leq \text{CGTHRESH}$. This rule is rather arbitrary, but it allows avoiding the expense of too many conjugate gradient iterates when an accurate solution of (2.8) is not required, namely away from the minimum on the current manifold. The overall work to approximately solve (2.8) is thus of the order of n times the number of conjugate gradient iterates used. Of course, it would be desirable to precondition the conjugate gradient iteration, but it is not clear how to do this since $Z^T G Z$ is not explicitly available.

Let us now consider the line search (2.9). There is no difficulty in using the line search of [14] unchanged, since all the necessary information is available. Usually one step of this line search obtains a sufficient decrease in F. This step requires estimating the zeros of all the inactive residuals in the direction along the line. Let us say that there are q inactive residuals, and let $\{z_j\}$, $j = 1,\ldots,q$, be the estimates of the zeros. It is explained in [14] how these values are compared with an estimate of the minimum of F along the line, supposing that F is smooth. If F does not appear to be smooth, because of the fact that the latter estimate lies beyond one or more of the $\{z_j\}$, it is necessary to compute the minimum of a piecewise linear function defined by the $\{z_j\}$. This computation amounts to finding a weighted median of q points, and is most easily done by a partial "Heapsort" of the values, which requires O(q log q) operations. However, it is possible to do this computation in O(q) time, although this is not worthwhile unless q is very large. See [1,15] for further details. In any case, this is not a significant factor at least in the latter stages of minimizing (1.3), since then q, the number of inactive residuals, is much less than m, the number of triangles.

Let us turn our attention to (2.7), by far the most difficult step. As already mentioned, the matrix A, when all columns are included, is highly rank deficient, and hence the Lagrange vectors of (2.4) are not uniquely defined. Consequently it is very difficult to tell whether a given point v, which is optimal on the current restricting manifold, is in fact optimal in the whole space. Furthermore, even if it were known that v is not optimal, it would be very difficult to decide which terms to delete from the active set to obtain a descent direction. This type of difficulty is called degeneracy in the context of linear programming and linearly constrained optimization. The only known methods guaranteed to overcome degeneracy require time exponential in the number of active terms, which is out of the question. We therefore have the unsatisfactory situation that we must terminate the algorithm with a "solution" that we know to be optimal on the final restricting manifold, but for which we are unable to verify optimality in the whole space.

This difficulty seems to be inherent in problem (1.3), because of the extreme built-in degeneracy. However, there are a number of positive remarks which can be made. First, assuming that the tolerances of [14] are chosen small enough so

that an accurate solution is demanded, note that there is no possibility of the algorithm terminating with too few residuals set to zero. The only danger is that too many of them may have been set to zero. Second, the situation is comparable to the well known problem of finding the global minimum of a nonconvex function. Several runs may be made, using different starting values or different choices of the parameters, and the lowest "minimum" found may be taken as the best candidate for the solution. Third, and most important, the numerical results of the next section indicate that, at least for a certain initial guess $v^{(0)}$, the "solutions" to which the algorithm converges in practice are very nearly optimal. Fourth, we mention a suggestion of Dax [6] at the end of the next section.

We conclude this section by describing an alternative approach to solving (1.3), based on ideas of [6,8] and others. This requires consideration of the hyperbolic approximating (HAP) function of [8], given by

$$\hat{F}(v) = \sum_{i=1}^{m} (\|r_i(v)\|^2 + \varepsilon_H^2)^{1/2} . \tag{3.5}$$

The function $\hat{F}$ is differentiable everywhere provided that $\varepsilon_H > 0$. The smaller the value of ε_H, the better $\hat{F}$ approximates F, but a consequence of this, of course, is that the smaller the value of ε_H, the worse is the conditioning of $\hat{F}$. The alternative method, then, simply requires minimizing $\hat{F}$ directly, using a straightforward Newton method. This has the advantage that the degeneracy is avoided and optimality of a point minimizing $\hat{F}$ can be verified. However, there are four serious difficulties with this approach. First, a minimum of $\hat{F}$, not F, is obtained. This may not be important, since (1.3) is derived from (1.1) by a discretization, but it means that a solution to (1.3) is not obtained. Second, if ε_H is small, the extreme ill-conditioning of $\hat{F}$ makes the convergence of a Newton method very slow. Third, each Newton step requires the solution of a linear system with much larger dimension than that of (2.8), since there are no zero residuals to reduce the dimension of the space. Especially in the last stages of the iteration, the dimension of (2.8) is much less than n, and consequently (2.8) can be approximately solved using fewer conjugate gradient iterates than a full Newton system requires. Fourth, if ε_H is small, the conditioning of the full Newton system is much worse than the conditioning of (2.8), and hence the convergence of the "inner" iteration can be expected to be much slower than that for (2.8). This consideration is separate from the convergence of the "outer" nonlinear iteration (the second point) and the dimension of the linear system (the third point).

This alternative algorithm, minimizing $\hat{F}$, is easily implemented in the same program that was prepared for the direct minimization of F. In particular, although the alternative algorithm does not allow zero residuals, the same linked list data structure is used to impose the boundary conditions and the constraint (1.3c). Thus the "full Newton" system is actually implemented in the form (2.8), but with the matrix Z having rank equal to the number of variables minus the number of constraints. This corresponds to a matrix $\bar{A}$ containing only the columns which describe the constraints, i.e. the 1×1 blocks for the Dirichlet conditions, and the last column for (1.3c).

NUMERICAL RESULTS

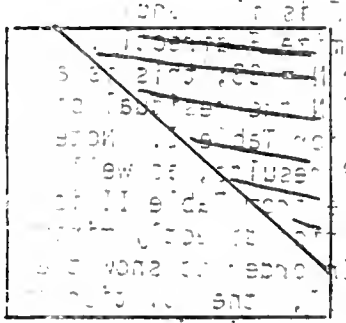
The results were obtained using Fortran on a VAX 11/780 at the Courant Mathematics and Computing Laboratory. Double precision arithmetic, i.e. approximately 16 decimal digits of accuracy, was used throughout. We restricted the experiments to Strang's model problem, i.e. (1.1) with $f \equiv 0$ and $c \equiv 1$, whose solution is shown in (1.2). Because of the symmetry, we actually solved the problem on the lower left quarter of the square shown in (1.2). Thus the domain was a unit square, with N mesh points in each direction, homogeneous Dirichlet boundary

conditions on the lower and left sides, and with the points on the other two sides constrained to match pairwise. This is conveniently implemented using the linked list data structure, with an initial configuration of one fixed list for each point on the lower and left sides, one variable list for each pair of points on the upper and right sides, and one variable list for the top right corner point and each interior point.

We chose the initial vector $v^{(0)}$ as follows, where i and j range from 0 (lower or left side) to N-1 (upper or right side):

$$v^{(0)}(x_i,y_j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \text{ (Dirichlet boundary conditions)} \\ N-1 & \text{if } i + j \geq N \\ i+j-1 & \text{otherwise} \end{cases}$$

This vector was then scaled to satisfy (1.3c). The corresponding piecewise linear function is constant, i.e. has zero residuals, in the shaded part of:



and is linear across the remainder of the grid except at the boundary. This choice of $v^{(0)}$ is a compromise between $v^{(0)}$ = constant in the interior, which has too many zero residuals, and v = linear, not constant, in the interior, which having no zero residuals causes Z^TGZ to be singular initially.

The results are summarized in Tables I and II. Table I gives the results of solving (1.3) when high accuracy is demanded. In this case the parameters of [14] were set as follows: $\varepsilon_{MCH} = 10^{-16}$, $\varepsilon_{ACT} = 10^{-4}$, $\varepsilon_{PG} = 10^{-6}$ (except 10^{-5} for N = 33), $\varepsilon_{LINE} = 10^{-12}$, $\eta = 0.9$. Of these, the most important is ε_{PG}, which specifies how small the norm of the projected gradient, $\|Z^Tg\|$, is to be made. The parameter ε_{ACT} determines when an attempt is made in (2.6) to make "almost" active residuals "exactly" active. This parameter may be reduced dynamically by the program; see [14] for details. The conjugate gradient iteration parameters were set as follows: CGTOL = 10^{-1}, CGTHRESH = 10^{-2}, and CGMAX as shown in the table. The other columns in the table specify N (the number of mesh points in each direction), the column rank of Z at the computed solution (i.e. the dimension of the final restricting manifold), the final value of F, ITER (the number of major iterates required), and CGITER (the total number of conjugate gradient iterates). The most interesting thing to note in Table I is the consistency of the final values of F determined by the various choices of CGMAX. The dependence of CGITER, the best measure of the total amount of work, on CGMAX, is also interesting although somewhat inconclusive.

The minimizing solution v is shown for the case N = 13 in Table III. Note that there are two regions where v is constant, the large one in the top right and the small one in the bottom left. The band of varying values between the two regions is, of course, the result of the discretization. One can see that the circular arc Γ of (1.2) is reflected in this band of varying values since, for example, $v(x_2,y_2) = 0.209$ is greater than $v(x_1,y_3) = v(x_3,y_1) = 0.149$.

In order to show how well the solution of (1.3) approximates the solution of (1.1) as N is increased, we display in Figure 1 graphs of the solution v plotted along the diagonal of the square. The results from Table I are shown for N = 9, 17 and 33. The step function shown is the analytic solution of (1.1). The distance from the mesh point (0,0) to the discontinuous jump in the solution, divided by the length of the diagonal, is $\rho(1 - 1/\sqrt{2}) \approx 0.155$, where ρ is given following (1.2). The analytic solution has a value $u = 1/2(1 - \rho^2 + \pi\rho^2/4) \approx 1.064$ in the constant positive region, with a corresponding minimal F value of $2-2\rho+\pi\rho/2 \approx 1.772$.

Table II summarizes results when a much less accurate solution of (1.3) is required. Here we show results both for the direct minimization of F and for the minimization of the HAP function $\hat{F}$, described at the end of Section 3. In the latter case, we set $\varepsilon_H = 10^{-1} \cdot (h^2/2)$, where $h = 1/(N-1)$, since the residual sizes are proportional to the areas of the triangles. If a much larger value of ε_H is used, the function $\hat{F}$ approximates F very poorly; if a much smaller value is used, the computation time becomes excessive. We set ε_{PG}, the tolerance on the projected gradient, to 10^{-1}. Again, a much smaller value of ε_{PG} leads to excessive computation time, since F is ill-conditioned. We therefore also used $\varepsilon_{PG} = 10^{-1}$ for the runs which minimize F directly. We also set $\varepsilon_{ACT} = 10^{-2}$ for these runs, except $\varepsilon_{ACT} = 10^{-4}$ for N = 33; this is actually relevant only for $N \leq 9$, since for larger values of N the residual sizes are smaller. The other parameters had the same values as for Table I. Note that it is F, not $\hat{F}$, which is shown in the column for the HAP results, as well as for the direct results. The interesting observation to make from Table II is that even when an inaccurate solution is required, the method which directly minimizes F has substantial advantages over the HAP method. In order to show the difference in the accuracy of the solutions from Tables I and II, the solution obtained from minimizing the HAP function when N = 17 is plotted in Figure 1.

Dax [6] has made an interesting suggestion regarding the use of the HAP function in the context of multifacility location problems. He suggests switching to the minimization of F only after reaching the minimum of F on the current manifold, as a way of potentially escaping from a nonoptimal point. If a lower value of F is obtained in the process, a switch can be made back to the minimization of F. This seems a useful idea, especially if $v^{(0)}$ is poorly chosen, e.g. $v^{(0)}$ = constant in the interior. In this case the direct algorithm would terminate immediately, but a switch to minimizing F produces a lower value of F without difficulty. However, this idea does not produce any benefits for the runs given in Table I, apparently because the direct minimization produced very nearly optimal results.

CONCLUDING REMARKS

In this paper we have confined our attention to the description of a method which obtains accurate solutions to (1.3), a discretization of (1.1). Some alternatives to (1.1) which we have not considered include allowing discontinuous elements (Johnson [11]) and solving the primal problem for which (1.1) is the dual (Strang [16]). Other relevant papers include [5,10,13,17,18]. We conclude with the remark that if a method such as the one described in this paper is to be used to obtain very accurate solutions to (1.1) or related problems, some sort of mesh refinement near the curve of jump discontinuity would probably be desirable.

ACKNOWLEDGMENTS

The author would like to thank Gene Golub for introducing him to the topic of this paper. Thanks also go to Robert Kohn and Olof Widlund for many helpful conversations, and to Gilbert Strang for his interest and comments.
This work was supported in part by NSF Grant MCS-8302021 and in part by the U.S. Department of Energy under Contract DE-AC02-76-ER03077-V.

[1] Bleich, C. and Overton, M. L., A linear-time algorithm for the weighted median problem, Computer Science Dept. Report No. 75, Courant Institute of Mathematical Sciences, New York (April 1983).

[2] Calamai, P. H., On numerical methods for continuous location problems, Ph.D. Thesis, Dept. of Systems Design, University of Waterloo, Waterloo, Ontario (1983).

[3] Calamai, P. H. and Conn, A. R., A stable algorithm for solving the multifacility location problem involving Euclidean distances, SIAM J. Scient. and Stat. Comp. 1 (1980) 512-526.

[4] Calamai, P. H. and Conn, A. R., A second-order method for solving the continuous multifacility location problem, in: G. A. Watson (ed.), Numerical Analysis (Proc. 9th Biennial Conf., Dundee, Scotland), Lecture Notes in Mathematics 912 (Springer Verlag, 1982), 1-25.

[5] Christiansen, E., Computation of limit loads, Report 2, Matematisk Institut, Odense University (1979).

[6] Dax, A., The use of Newton's method for solving Euclidean multifacility location problems, Hydrological Service, P. O. Box 6381, Jerusalem (1983).

[7] Dembo, R. S. and Steihaug, T., Truncated Newton algorithms for large scale unconstrained optimization, Math. Programming 26 (1983) 190-212.

[8] Eyster, J. W., White, J. A. and Wierwille, W. W., On solving multifacility location problems using a hyperboloid approximation procedure, AIIE Transactions 5 (1973) 1-6.

[9] Fleming, W. H., Functions with generalized gradients and generalized surfaces, Annali di Matematica 44 (1957) 93-103.

[10] Grierson, D. E., Collapse load analysis, NATO-ASI Lecture Notes on engineering plasticity by mathematical programming, University of Waterloo, Waterloo, Ontario (August 1977).

[11] Johnson, C., Error estimates for some finite element methods for a model problem in perfect plasticity, Report, Chalmers University of Technology, Gothenburg, Sweden (1981).

[12] Kuhn, H. W., On a pair of dual nonlinear programs in: J. Abadie (ed.), Nonlinear Programming (North-Holland, Amsterdam, 1967) 38-54.

[13] Matthies, H., Problems in plasticity and their finite element approximation, Ph.D. Thesis, Mass. Inst. of Tech. (1978).

[14] Overton, M. L., A quadratically convergent method for minimizing a sum of Euclidean norms, Math. Programming 27 (1983) 34-63.

[15] Reiser, A., A linear selection algorithm for sets of elements with weights, Information Processing Letters 7 (1978) 159-162.

[16] Strang, G., A minimax problem in plasticity theory, in: Nashed, M.Z. (ed.), Functional Analysis Methods in Numerical Analysis, Lecture Notes in Mathematics 701 (Springer Verlag, 1979).

[17] Strang, G. and Matthies, H., Mathematical and computational methods in plasticity, presented at Sept. 1978 IUTAM Conf. on Variational Methods in the Mechanics of Solids, Evanston, Illinois (1979).

[18] Yang, W. H., A practical method for limit torsion problems, Computer Meth. in Appl. Mech. and Eng. 19 (1979) 151-158.

[19] Rockafeller, R.T., Convex Analysis (Princeton University Press, 1970).

TABLE I

Results of Minimizing F Accurately ($\varepsilon_{PG} = 10^{-6}$)

N	CGMAX	RANK(Z)	F	ITER	CGITER
5	5	3	2.5377626	7	14
	10	3	2.5377626	7	14
9	5	9	2.2305853	24	92
	10	9	2.2305853	22	91
13	5	19	2.1199518	59	231
	10	20	2.1198360	57	383
	15	20	2.1198360	40	286
	20	21	2.1198359	41	400
17	5	35	2.0629985	122	623
	10	36	2.0629969	78	496
	15	41	2.0629921	57	533
	20	41	2.0629921	85	919
33	5	140	1.9759079	562	5133
($\varepsilon_{PG}=10^{-5}$)					

TABLE II

Results of Minimizing Both F and the HAP Function $\hat{F}$ Inaccurately ($\varepsilon_{PG}=10^{-1}$)

		Results of Minimizing F Directly				Results of Minimizing the HAP Function $\hat{F}$			
N	CGMAX	RANK(Z)	F	ITER	CGITER	RANK(Z)	F	ITER	CGITER
5	5	3	2.542	4	5	12	2.553	7	30
	10	3	2.542	4	5	12	2.549	5	30
9	5	9	2.235	10	18	56	2.260	25	120
	10	9	2.239	10	29	56	2.252	12	103
13	5	54	2.184	8	12	132	2.169	44	215
	10	27	2.128	17	57	132	2.155	18	163
	15	25	2.122	26	126	132	2.146	16	159
	20	18	2.121	20	110	132	2.142	11	200
17	5	79	2.099	20	43	240	2.131	48	231
	10	66	2.083	22	95	240	2.102	30	278
	15	55	2.077	22	134	240	2.104	20	207
	20	62	2.082	15	118	240	2.100	16	226
33	5	438	2.060	24	79	992	2.101	46	197
	15	370	2.031	92	874	992	2.051	33	385

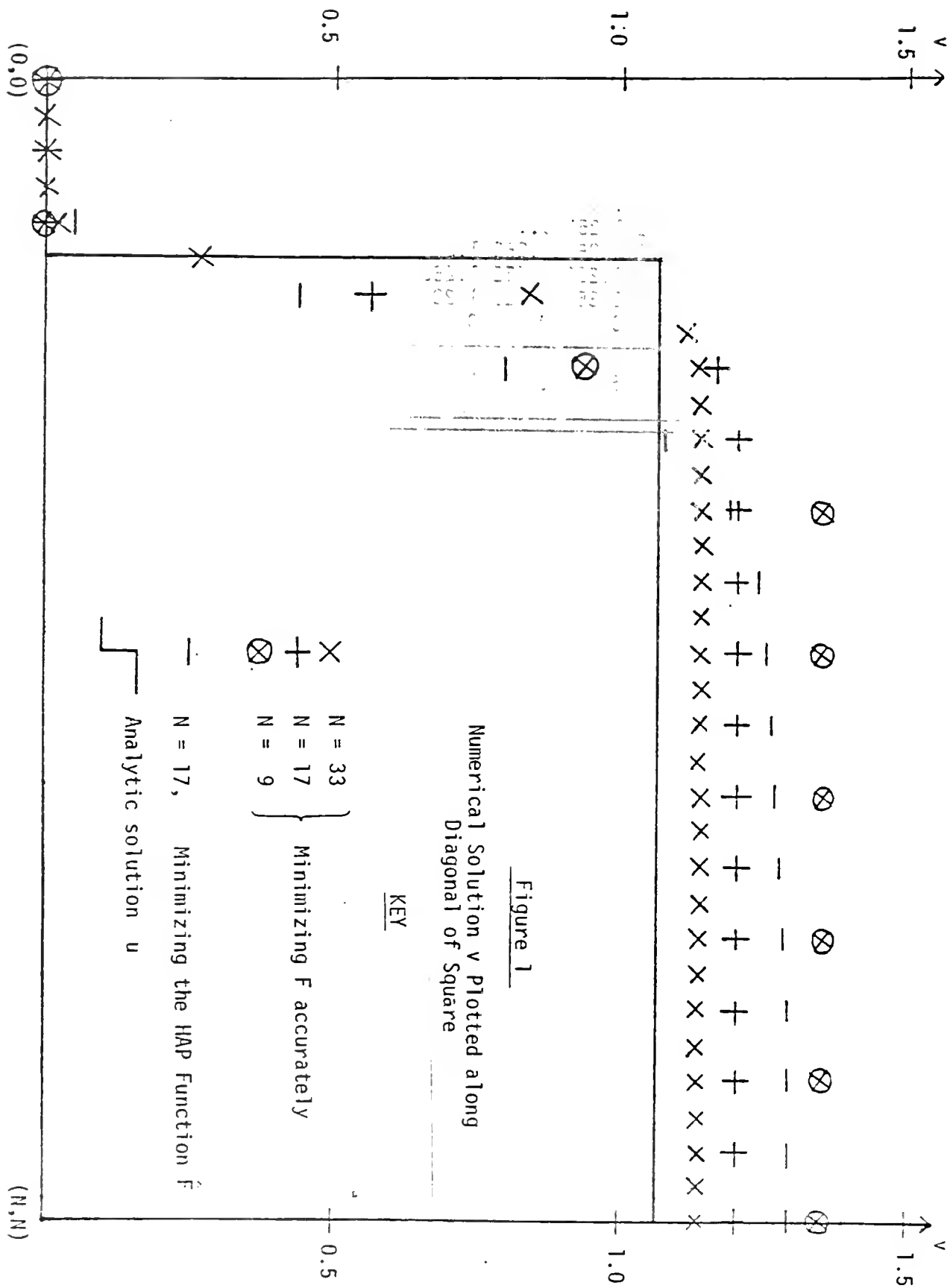


Figure 1

Numerical Solution v Plotted along Diagonal of Square

TABLE III

Numerical Solution $v(x_i, y_j)$ for N = 13 ($\varepsilon_{PG} = 10^{-6}$, CGMAX = 10,15,20)

12	0	1.252	1.252	...								...	1.252
11	0	1.252	1.252	...									⋮
10	0	1.252	1.252	...									⋮
9	0	1.252	1.252	...									
8	0	1.252	1.252	...									
7	0	1.232	1.252	1.252	...								
6	0	1.123	1.252	1.252	...								
5	0	0.857	1.229	1.252	1.252	...							
4	0	0.482	1.077	1.242	1.252	1.252	...						
3	0	0.149	0.686	1.120	1.242	1.252	1.252	1.252	...				
2	0	0	0.209	0.686	1.077	1.229	1.252	1.252	1.252	1.252	1.252	1.252	1.252
1	0	0	0	0.149	0.482	0.857	1.123	1.232	1.252	1.252	1.252	1.252	1.252
0	0	0	0	0	0	0	0	0	0	0	0	0	0
j / i	0	1	2	3	4	5	6	7	8	9	10	11	12

c.2
NYU
Comp. Sci. Dept.
TR-100 Overton
AUTHOR
Numerical solution of a
TITLE
model problem from ...

DATE DUE	BORROWER'S NAME